Labo 4 - Web Scraping / Crawling

Hektor Misplon

Inhoudstafel

Inleiding

Repo Structure

```
├──assets
│   ├──docs
│   │   └──examples (example tutorials)
│   └──images
└──examples (example code)
```

Theorie

Wat is web scraping?

Web scraping is het software-matig uitlezen & verwerken van data.
Met andere woorden is web scraping een geautomatiseerde manier om gericht op zoek te gaan naar info.
Een web crawler gaat op een bepaalde pagina op zoek naar info om daarna de data uit deze sites te extraheren.
Deze data wordt verwerkt & geparsed in het gewenste formaat

Wat is web crawling?

Data crawling slaat specifiek op het downloaden/indexeren van webppagina's, dit wordt dan ook vooral op grote schaal gedaan.
Hierbij is het voorkomen van dubbele informatie een belangrijk gegeven.

Hoe werkt web scraping & crawling?

Web scraping begint met het gebruik van zogenaamde spiders die de HTML van relevante webpagina's gaan ophalen.
Nadat deze is opgehaald kan hierin op zoek gegaan worden naar de nodige informatie om die dan uiteindelijk in een bepaald formaat (al dan niet in een database) te gaan opslaan.

Waarvoor wordt web scraping & crawling zoal toegepast

Web Scraping kan heel wat informatie verschaffen, daarom heeft het dan ook tal van toepassingen.
Om maar enkele voorbeelden te geven:

- Het kan gebruikt worden in e-commerce voor het ophalen van producten met hun corresponderende prijzen om zo al dan niet in te spelen op de markt.
- Veel bedrijven gebruiken web scraping om info te verwerven bij het maken van strategische keuzes
- Wetenschappers hebben ook baat bij de talrijke data die kan verworven worden door gebruik van web scraping

Scraping richtlijnen

Bij het kiezen van een webpagina om te scrapen zijn er een aantal voorwaarden verbonden.
Zo kan de eigenaar van de webpagina de toestemming (deels) weigeren voor het scrapen, wanneer dit genegeerd wordt kunnen we de toegang tot de pagina verliezen.
De meeste sites hebben dan ook hun voorwaarden voor het scrapen van hun site duidelijk uitgeschreven, deze voorwaarden kunnen we steeds terugvinden in de robots.txt file.
De robots.txt file vinden we steeds terug door het /robots.txt pad toe te voegen aan de domeinnaam van de site (bv. www.voorbeeld.com/robots.txt).

robots.txt

Check de robots.txt file vooraleer je een site gaat scrapen
Terug te vinden vinden op deze manier: https://www.tescrapendomein.com/robots.txt

Mogelijke richtlijnen

- Volledige toegang (alle paginas kunnen gecrawled worden)

```
User-agent: *
Disallow:
```

- Geen toegang (geen enkele pagina mag gecrawled worden)

```
User-agent: *
Disallow: /
```

- Specifieke toegang (alleen de aangegeven pagina's mogen gecrawled worden)

```
User-agent: *
Disallow: /folder/
User-agent: *
Disallow: /file.html
```

- Crawl interval limiet - onderstaand vb. = 11 seconden (er kan een pauze van een bepaald aantal seconden opgelegd worden tussen het crawlen van de pagina's, vooral tegen het overbelasten van de servers)

```
Crawl-delay: 11
```

- De visit time bepaalt de periodes waarin gecrawled mag worden - onderstaand voorbeeld: 1:00 - 5:00 UTC

```
Visit-time: 0100-0300
```

- De request rate bepaalt het het aantal pagina's die gecrawled mogen worden om de 10 seconden

```
Request-rate: 1/10
```

Web scraping stapsgewijs

1. Vooraleer we gaan scrapen hebben we een webpagina nodig om te scrapen, hierbij houden we rekening met de voorziene richtlijnen voor het scrapen in het robots.txt bestand. Indien dit bestand aanwezig is vinden we het terug door naar het pad /robots.txt in de domeinnaam van de site te gaan. (www.voorbeeld.com/robots.txt)
2. Nadat we de te scrapen webpagina bepalen gaan we de structuur van de webppagina analyseren. Hierbij kijken we naar de HTML van de pagina (paginabron inspecteren), hieruit kunnen we meteen afleiden welke HTML elementen relevant zijn. We kunnen ook kijken naar de classes & id's die aanwezig zijn die bruikbare informatie bevatten.
3. Na deze stappen kunnen we gaan scrapen, hiervoor kiezen we eerst onze werkwijze/tools.
4. Nu kunnen we (mbhv de tools) de code schrijven voor onze scraper
5. Nadat we de info verzamelen moeten we de tekst die we nodig hebben nog uit deze data gaan filteren zodat we deze in een bepaald formaat kunnen gaan structureren

6. Nu we de gestructureerde data hebben kunnen we deze opslaan in een database

Python Web Scraping tools

Vooraleer we van start gaan overloop ik de tools waarmee we in contact komen bij web scraping of web crawling in python.
Zo krijgen we een goed overzicht van de verschillende tools die we beschikbaar hebben en kunnen we onze focus meteen op de geschikte tools voor onze noden leggen.

Scrapy (framework)

+ Voordelen

- alles-in-één oplossing voor web scraping
- dit framework is gebouwd op **Twisted** library [2] (= event-driven, open-source & asynchrone library voor netwerken)
  - efficiënt
  - asynchroon
- flexibel, veel mogelijkheden
- bepaalde gelijkenissen met het **Django** framework, kan handig zijn voor mensen met ervaring in Django

+ Nadelen

- stijle leercurve, minder toegankelijk voor beginners
- windows installatie vereist wat extra stappen en verloopt niet altijd even vlot

Support

- [x] Python 2
- [x] Python 3

Urllib2 - HTTP module (reading/opening web pages)

+ Voordelen

- inbegrepen in de standaard library van Python

+ Nadelen

- vroeger populairder dan nu, de gelijkaardige tool genaamd 'Requests' neemt nu de overhand

Support

- [x] Python 2
- [x] Python 3

Requests - HTTP module

+ Voordelen

- een zeer populaire library voor Python
- goede documentatie

+ Nadelen

- komt niet standaard met Python

Support

- [x] Python 2
- [x] Python 3

BeautifulSoup

BeautifulSoup gebruiken we om data uit te laden vanuit de 'data points' van een geladen pagina
De naam komt van een term die gebruikt word voor een slechte opmaakstructuur, die 'soup' kan genoemd worden.

+ Voordelen

- goed gedocumenteerd
- veel goede voorbeelden beschikbaar online
- gemakkelijke installatie
- gemakkelijk in gebruik

Support

- [x] Python 2
- [x] Python 3

lxml

lxml is erg gelijkaardig met BeautifulSoup. Het kan minder toegankelijk zijn voor beginners, maar heeft wel de betere hand in performantie.

- rijk aan features
- snel
- maakt efficiënt gebruik van het geheugen
- er zijn betere voorbeelden te vinden dan de voorbeelden vanuit de documentatie

selenium

Selenium is een tool die helpt bij de automatisering voor het testen van web applicaties

+ Voordelen

- toegankelijk voor beginners
- kan beter om met pagina's waarvan de structuur erg afhankelijk is van JavaScript
- goede documentatie

Support

- [x] Python 2
- [x] Python 3

Demo & tutorials

BeautifulSoup Installation

BeautifulSoup installeren we mbhv volgend commando in python 2 of 3 mbhv pip.

```
$ pip install beautifulsoup4
```

Scrapy Installation

Linux

Installeer de dependencies vanuit terminal

```
sudo apt-get install python-dev python-pip libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-
```

Install Scrapy from terminal w/ pip

```
sudo pip install scrapy
```

Mac

Installeer Scrapy vanuit terminal mbhv pip

```
sudo pip install scrapy
```

Windows

1. Eerst installeren we **Anaconda**
    - Ga naar https://www.anaconda.com/distribution/
    - Zorg dat je de juiste installer gebruikt (e.g. 64-bit)
2. Gebruik volgend commando om **Scrapy** te installeren

    ```
    pip install scrapy
    ```

3. (optioneel) indien voorgaande niet werkte kun je het volgende commando uitproberen

    ```
    conda install -c conda-forge scrapy
    ```

4. Het is mogelijk om deze error te krijgen
   // TODO: screenshot
   Indien je deze error krijgt kun je dit oplossen door de Visual Studio Build Tools te installeren
   // TODO: screenshot fix

Scrapy basis

CLI (command line interface)

Om te controleren welke versie van Scrapy je geinstalleerd hebt gebruik je volgend commando

```
~$ scrapy version
```

Alle voorbeelden werden gemaakt met Scrapy 1.6.0 dus om onvoorziene errors te voorkomen raad ik aan om dezelfde versie te installeren

Om alle beschikbare commandos op te lijsten gebruik je volgend commando

```
$ scrapy
```

Ik raad aan om de volgende commandos eens te doorlopen om een snel overzicht te krijgen van de Scrapy CLI.
Indien je meer te weten wil komen over de Scrapy CLI kun je de **Scrapy docs** raadplegen.

Output:

```
Scrapy 1.6.0 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench         Run quick benchmark test
  fetch         Fetch a URL using the Scrapy downloader
  genspider     Generate new spider using pre-defined templates
  runspider     Run a self-contained spider (without creating a project)
  settings      Get settings values
  shell         Interactive scraping console
  startproject  Create new project
  version       Print Scrapy version
  view          Open URL in browser, as seen by Scrapy

  [ more ]      More commands available when run from project directory

Use "scrapy <command> -h" to see more info about a command
```

Een nieuw Scrapy project aanmaken

1. Maak een nieuwe folder aan in de directory waar je een nieuw Scrapy project wil aanmaken. Navigeer in deze folder

   ```
   $ mkdir scrapy-projects
   $ cd scrapy-projects/
   ```

2. Start een niew Scrapy project genaamd 'first-project'

   ```
   /scrapy-projects$ scrapy startproject first-project
   ```

3. Navigeer naar de folder die aangemaakt werd door Scrapy

   ```
   /scrapy-projects$ cd first-project
   ```

4. Check het aangemaakte project om zeker te zijn of Scrapy een nieuw project aanmaakte. Wanneer je het volgende commando runt zou je ook bovenaan de naam van je project moeten zien

```
$ scrapy
```

5. Genereer een nieuwe spider, in dit commando geef je de naam voor de nieuwe spider door en de domeinnaam van de webpagina die je wil scrapen

```
~/scrapy-projects/first-project$ scrapy genspider first-spider example.com
```

6. Lijst de spiders voor dit project op met volgend commando

```
$ scrapy list
```

Als alles goed gegaan is zou je de structuur van je project er ongeveer zo uit moeten zien

```
├──scrapy.cfg
└──first-project/
    ├──__init__.py
    ├──items.py
    ├──middlewares.py
    ├──pipelines.py
    ├──settings.py
    └──spiders/
        ├──__init__.py
        └──first-spider.py
        ...
```

Het scrapy.cfg bestand in de root directory bevat de Scrapy instellingen voor dit project.

Examples

De voorbeelden bevatten commentaar die de code duidelijk illustreert.
Ga naar voorbeelden

Best Practices

- Respecteer de voorwaarden die door de eigenaar van de webpagina zijn opgelegd (robots.txt)
- Rescecteer de servers van de webeigenaar & leg limieten/intervallen op bij het scrapen van de webpagina

User agent, IP & proxy rotation

Voorkom geblokkeerd te worden door het gebruik van verschillende User Agents & IP adressen.
De webpagina wil de toegang voor gewone gebruikers niet beperken, op deze manier kunnen we onze crawlers op verschillende gewone gebruikers laten lijken.
In een van de voorbeelden is een proxy-crawler terug te vinden die verschillende proxies ophaalt, deze kunnen hiervoor toegepast worden.